High-Density Network Flow Monitoring

Petr Velan
CESNET, z.s.p.o.
Zikova 4, 160 00 Praha 6, Czech Republic
petr.velan@cesnet.cz

Viktor Puš
CESNET, z.s.p.o.
Zikova 4, 160 00 Praha 6, Czech Republic
pus@cesnet.cz

***Abstract*—Monitoring of high-speed networks is becoming a resource intensive task. There are dedicated flow monitoring probes built with commodity hardware support up to 10 G links, but multiple 10 G or even 100 G optical networks are being used for transport networks and a data center connectivity. Running and maintaining many separate probes is uneconomical and time-consuming. Therefore, we explore the possibility to facilitate network interface cards (NICs) with multiple 10 G interfaces to build probes which can replace many existing boxes, leading to reduced management and operational costs. The monitoring performance is critical for such a high-density solution. We use two custom-built, FPGA-based NICs, each with eight 10 G interfaces to test current CPU limits and to propose improvements for the near future commodity NICs.**

I. Introduction

A current commodity hardware for high-speed network traffic flow monitoring usually consists of a PC with one or two network interface cards (NICs) with a 10 G Ethernet interface. The role of the hardware is merely to capture the data from the network. The hardware setup is supplemented by open- or closed-source software performing the subsequent steps of the flow monitoring process, most commonly packet header parsing, flow cache management and flow record export using the NetFlow [1] or IPFIX [2] protocol.

While one can deploy multiple such probes for monitoring of multiple lines, it may not be the best option because of an increased power consumption, a large rack space footprint and a general complexity of the monitoring system management. Our work therefore focuses on exploring the scalability of this concept in a single PC from the performance point of view. There are obviously concerns about various possible bottlenecks of a single PC setup. To name a few: throughput of the NIC, system bus (i.e. PCI Express) and PC memory subsystem, performance of a single CPU core, overhead of potential inter-core or inter-CPU communication.

By using custom-built NICs, we are able to perform experiments with the speeds beyond what is available in the commodity hardware market. The programmability of our FPGA-based NICs allows us to examine the effect of various additional hardware features, such as a packet header parsing or a packet shortening. We suggest that some of these features should be included in the next generation of commodity NICs to aid the performance of future network traffic monitoring systems.

The aim of this paper is to identify bottlenecks and potentials for new features of current and near-future monitoring systems, as well as to demonstrate limitations of current CPUs and generally the PC architecture in the particular use case of network flow monitoring. We test a monitoring setup that can be achieved using commodity NICs, then we present impacts of the features provided by our NIC. Last but not least, we compare two different CPUs in the same conditions to determine the influence of CPU choice on the monitoring performance.

The paper is structured as follows: Section II presents a choice of notable previous works in this field. Section III describes our monitoring architecture and how it is different from current commodity NICs. Section IV explains our experiments and methodology, while Section V presents the obtained results. The last Section VI draws conclusions from the results.

II. Related Work

We provide a brief overview of progress in network monitoring over the last decade. The list of works is by no means complete, nevertheless, it should outline the achievements in this field so far.

1 G networks were state of the art in 2003. Luca Deri in [3] shows that it is possible to create a 1 G flow probe based on a commodity hardware. He uses libpcap library to receive packets from the NIC and shows that it is possible to capture the packets at line speed. Therefore, the lack of performance discovered in a comparable systems is caused by an ineffective processing software.

Degioanni and Varenni in [4] introduce the concept of using customized NICs for achieving better monitoring performance in 2004. The main contribution of their work is a design of a device driver for customized NIC, which can distribute the packet processing load to multiple CPU threads. Although they do not achieve 10 G speeds, the methods they describe are still used nowadays.

Article from Clegg et al. [5] published in 2008 comprehensively describes practical and legal aspects of monitoring of a 10 G research network. Authors also provide several data sets containing captured packet headers as a part of their contribution.

Braun et al. [6] in 2010 compare commodity hardware based packet capturing solutions. Authors note that the software and commodity hardware are advanced enough to support 10 G packet capture solutions. The paper provides a detailed description and evaluation of techniques that help to achieve a high-speed packet capture. It describes improvements made to various libraries and NIC drivers that aim to remove several data processing bottlenecks.

The work of Fusco and Deri [7] published in 2010 introduces a design of packet capture solution fully utilizing multicore systems. They use PF_RING [8] with TNAPI drivers to avoid unnecessary buffer allocations and to utilize packet polling, which mitigates problems with too many interrupts.

Antichi et al. [9] propose to use NetFPGA platform for passive network measurement. They note that without hardware support, the packet timestamps have poor accuracy. The use of FPGA allows to timestamp, filter or trim the packets, which saves CPU processing time and allows to achieve higher packet rates. However, their work is targeted only on 1 G environment.

Use of commodity cards with advanced features like timestamping or packet filtering is proposed by Deri et al. [10] in 2013. Authors use these features to lessen the load on the CPU. They also utilize the capability of these cards to store data in multiple buffers, therefore exploiting multi-core architecture of their system.

Another recent work by García-Dorado et al. [11] provides overview of different commodity based packet capture solutions. The paper describes techniques used by various researches to achieve 10 G packet rates in detail. Authors explain features of current NICs, NUMA architecture, OS network stack as well as approaches to utilize these concepts and overcome performance bottlenecks.

A lot of work was done in the field of high-speed packet capture solutions. A recent survey of Hofstede et al. [12] describes the current state of flow monitoring systems including both open source and enterprise products.

Recent announcement of Intel XL710 NICs [13] represents an evolution of commodity hardware towards the concepts presented further in this paper. The NICs support acceleration features such as TCP checksum offload to decrease the CPU usage.

III. Monitoring Architecture

This section describes our monitoring setup from hardware to software. We briefly introduce our custom-built FPGA-based NIC architecture. We utilize these NICs to incorporate multiple 10 G interfaces in one box.

A. Hardware

We use the COMBO-80G card [14] to receive the network traffic. The card is equipped with two QSFP+ cages, which can be set to 4×10 G Ethernet mode each, thus creating eight 10 G interfaces in total. The packets undergo a configurable processing in the FPGA and are sent to the host RAM via the PCI-Express gen3 x8 bus. Theoretical throughput of this bus is 64 Gbps, but due to the protocol overhead, the real-world throughput is slightly higher than 50 Gbps. Figure 1 shows the functional scheme of the described hardware.

The FPGA firmware of COMBO-80G has several features that set it apart from conventional NICs and help to improve general throughput when receiving packets. The card automatically assigns a 64-bit timestamp to each packet at the moment of reception. The timestamp has a resolution of one nanosecond, which is better than what can be achieved by assigning it later by the software application. Also the processor load associated with the timestamp generation is removed.

Another feature is a configurable packet trimming. The card can be set to trim the packets to a predefined length, thus saving the PCI Express and memory subsystem bandwidth, most notably for long packets. This feature is clearly intended for the purpose of flow monitoring, since the relevant information (packet header) is at the beginning of each packet.

Further extension of this feature leads to packet parsing directly by the card and transferring only the parsed packet header fields to the host RAM. In addition to bandwidth saving, the processor load is also reduced, because it no longer needs to perform the packet parsing operation. The parsed fields are sent in the so-called Unified Header (UH) format which has fixed structure and thus removes the need for complicated parsing conditions in the corresponding processor code.

To better utilize current multicore CPUs, the firmware features an option to distribute the packets into eight independent DMA channels. Target channel number of each packet is computed by hashing several fields of the packet header, such as IP addresses, protocol, port numbers. This ensures that there is a consistent mapping of network flows to the DMA channels, so that the software threads always see complete flows. In common traffic with large number of flows, the traffic is evenly distributed among the DMA channels.

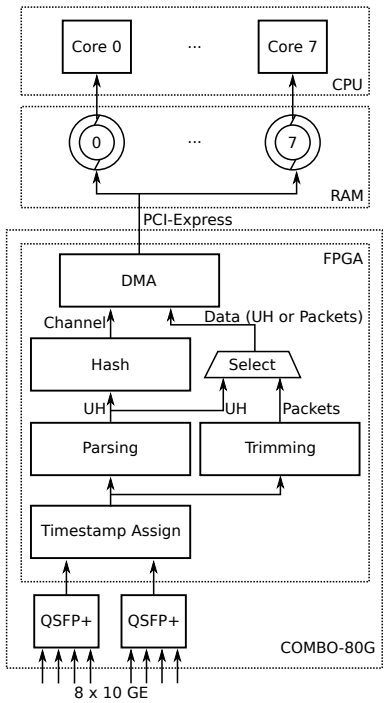

Figure 1. Hardware scheme of single card setup.

B. Software

The DMA transfers themselves are simplified and optimized to bypass an OS kernel network stack. Large ring buffers are allocated in RAM during the OS driver initialization, and the packets are then uploaded to these buffers by the card almost autonomously. The only communication between the

driver and the card is an exchange of buffer start and end pointers (which mark an empty and free space in the buffer), and configurable interrupts generated by the card. The OS driver never touches or even copies the packets, it only maps portions of the buffers to userspace applications. This way the overhead of software processing is minimized and maximum CPU time is left for the application itself. Directly mapping the ring buffers memory to userspace also ensures that the data are copied only once, from the NIC to the RAM. This decreases the load of the memory subsystem, which helps to increase the overall performance.

Data from the ring buffers are processed by a flow exporter. We use FlowMon exporter [15] software which utilizes a multithreaded design to distribute the computational load on multiple CPU cores. The flow exporter architecture is shown in Figure 2. Input threads read packets from ring buffers and parse L2 to L4 headers to flow records. These flow records are passed to a flow cache which performs flow record aggregation. Expired flow records are exported using single unifying thread, which accepts the flows from all flow caches. Single thread is enough for this task, since is is much less performance demanding.

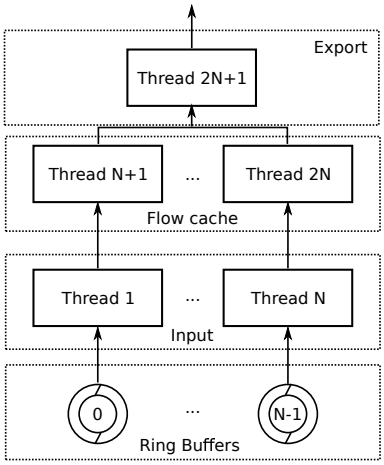

Figure 2. Flow exporter multithreaded architecture.

IV. Methodology

This section describes the setup for our experiments. We present our testbed as well as network traffic data that were used to measure the monitoring performance.

A. Testbed Setup

The testbed consists of two devices. The first is the flow monitoring probe and the second is a packet generator which generates traffic that is measured by the probe.

We use Dell PowerEdge R720 server with two Intel Xeon E5-2670 v1 CPUs as the flow monitoring probe. Each CPU features eight physical cores (16 with hyperthreading), 2.6 GHz operating frequency (3.0 GHz to 3.3 GHz in Turbo mode) and 20 MB of cache. Maximum TDP of each CPU is 115 W. The memory controller has throughput of 51.2 GB/s. Each CPU has four RAM modules available, running at 1600 MHz with the capacity of 8 GB per module (64 GB total). Two COMBO-80G cards are used to receive and process packets. The operating system is standard Scientific Linux release 6.5 running kernel version 2.6.32-431. The whole setup is a 2U standard rack mount PC.

Since the probe has two CPUs, we need to consider NUMA architecture specific setup. Each CPU has a directly accessible portion of RAM, which corresponds to the four physical RAM modules associated with the CPU. Accessing the memory of the other CPU is more costly, since the data need to be passed between the processors using QuickPath Interconnect bus (QPI) [16]. Each PCI-Express bus is also connected to one CPU. This CPU receives the interrupts of connected devices and the devices can write directly to the associated memory using DMA transfers. Therefore, the optimal setup is to have each COMBO-80G card on the PCI Express bus connected to different CPU. The memory allocated by the drivers for the NIC should belong to the same CPU and the flow exporter should also run on this particular CPU. This way, we can almost completely avoid QPI communications when processing the network traffic, which leads to higher performance. However, current version of the NIC drivers does not support NUMA specific memory allocation, which causes inefficient memory access through the QPI bus. The impact of this deficiency is described in Section V.

We run one instance of flow exporter on each physical CPU. This allows each exporter to use the memory of this CPU for the flow cache and therefore avoid to accessing the flow cache data through QPI.

B. Data Generator Setup

The test traffic is generated by Spirent TestCenter hardware generator at the speed of 10 Gbps and is replicated to all sixteen input ports. Since the data on all 10 G ports are the same, we use interface numbers in the flow creation process to ensure that the same packets from multiple interfaces will create different flows. It also ensures that the timestamps seen by flow the exporter are monotonous.

We use an artificial traffic pattern for the flow monitoring performance measurement. This approach has several advantages. Firstly, we can test the worst case scenario using short packets to achieve the highest packet per second ratio. Secondly, it is easy to repeat the tests in another laboratory. Moreover, it is infeasible to simulate a real traffic using packet generators, especially at 10 Gbps speed. To compensate for different number of concurrent flows and real traffic distribution of packets in flows, we repeat the tests with several different flow counts. All generated data are simple UDP packets, flows are created by permuting parts of IP addresses. The results provided in Section V are averaged from several measurements taken for each scenario.

Table I shows combinations of packet sizes and flow counts used in our measurements. Corresponding packet and bit rates are also shown. Various bit speed is caused by the Ethernet protocol overhead, which is lower for longer packets. All values are given for a single 10 G packet generator. Since the traffic is repeated to every input interface, the actual flow count and traffic rates sent into our monitoring device are 16 times higher.

Packet size	Flow count	Packets/s	Bits/s
64 B	128	14880545	7618839040
64 B	16384	14880545	7618839040
64 B	131072	14880545	7618839040
128 B	128	8445715	8648411900
128 B	16384	8445715	8648411900
128 B	131072	8445715	8648411900
256 B	128	4528861	9275108100
256 B	16384	4528861	9275108100
256 B	131072	4528861	9275108100
512 B	128	2349560	9623786500
512 B	16384	2349560	9623786500
512 B	131072	2349560	9623786500

Table I. COMBINATIONS OF PACKET LENGTHS AND FLOW COUNTS USED IN THE TESTS.

The flow count has high impact on the flow cache performance. With the increasing number of active flow records, the memory is accessed more randomly and the CPU experiences more cache misses, which results in higher latency of memory access and therefore in lower performance. It is difficult to estimate the number of flows to simulate from real network, since the real flow records are not updated periodically and the number of active flows in the flow cache at any given moment depends heavily on active and inactive flow cache timeouts and other software settings. Therefore, we test several different options to estimate the influence of flow count on the overall performance. We believe that the highest number of flow count used in our tests is more performance-demanding than real traffic in most networks.

V. RESULTS

The results of our experiments are presented in this section. We show the measurement performance in a setup achievable by commodity NICs, then we present the improvements achievable by our NICs. Moreover, we show a difference in performance for two different CPUs. We use packets per second as a measure for system performance. Bytes or bits per second can be gained easily by multiplying the number of packets per second by their respective sizes.

A. Basic Performance

First, we have measured the basic performance of the flow exporter on full packets. Figure 3 shows packet rates separately for each of the cards. We group the rates for the same packet lengths together. Each group has three pairs of columns. Each color represents a different number of flows per interface. Shades of the color differentiate the NICs. The *Maximum Ethernet* column is the highest achievable aggregated throughput of all 16 Ethernet input interfaces. The *Maximum PCIe* column has been measured by counting the packets received by the simplest possible software application (a packet counter). This way, we get an upper limit on number of packets, that the flow exporter can receive via the system bus.

We plot the CPU utilization in Figure 4. The interpretation of the graph is almost the same as for Figure 3 with the exception that values for both cards are summed up and the utilization is shown for different flow exporter threads instead.

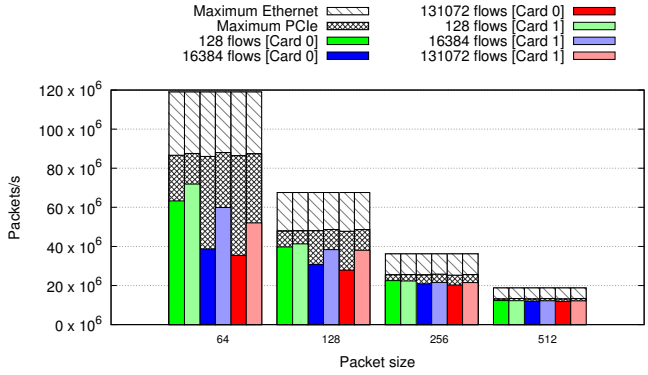

Figure 3. Full packet processing performance in packets/s.

The darker color represents the input thread and the lighter color marks the values for the flow cache thread.

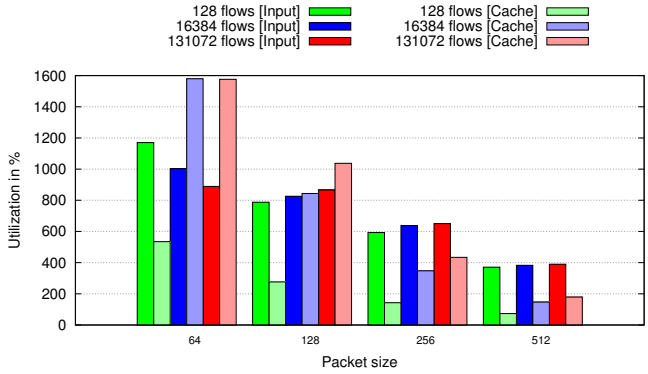

Figure 4. Full packet processing CPU utilization.

There are several important observations that can be made from these two graphs. The performance impact of number of flows is significant, especially for short packet lengths. This is caused by a high number of updates that must be performed in the flow cache. For a smaller number of flows, the CPU can keep a larger portion of the flow records cached in a faster memory. As the number of flows increases, the memory is accessed more at random, which causes more cache misses and eventually a performance decrease. Figure 4 clearly shows, that the utilization of the flow cache thread rises with number of flows for all packet lengths.

Performance of the second card is always higher than that of the first card. We attribute this to the device driver, which allocates packet buffers without considering the heterogeneity of the NUMA architecture. In our case, the first card and the corresponding flow exporter always access the RAM through the QPI bus using memory subsystem of the other CPU, which decreases the performance.

Although it seems that the performance is decreasing for longer packets, this depends on the point of view. The number of bytes per second is actually increasing for longer packets. Since less packets need to be processed to achieve the same throughput, the CPU utilization decreases. However, more data are processed, which requires higher memory controller throughput. Therefore, for longer packets, the performance is not hindered by insufficient CPU frequency, but by high

memory bus utilization.

B. Hardware Accelerated Performance

Secondly, we have measured the performance using hardware acceleration. Two different acceleration methods have been used. The first method is to set up the COMBO-80G cards to trim the packets to 64 bytes. This allows to keep the processing speed the same for all packet lengths. The second method uses internal parser of the NIC and sends only a predefined data structure, called Unified Header, with parsed information to the software. The main advantage is that the flow exporter does not have to parse the packet headers and therefore saves some CPU time. The disadvantage of both methods is that the flow exporter cannot perform any payload dependent analysis.

Figure 5 shows the performance comparison of the processing of full packets, trimmed packets and unified headers. We plot the graphs for 16 384 flows per interface, but the results are very similar for other flow counts. Note the throughput of PCI Express bus. Using the Unified Headers decreases the number of bytes that must be transferred for each packet. Therefore the throughput is higher even for the shortest packets.

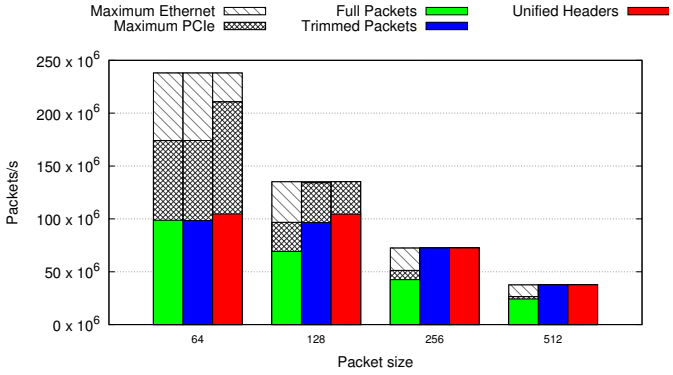

Figure 5. Packet processing performance comparison in packets/s for 16 384 flows per interface.

Using trimmed packets does not achieve any improvements for the shortest packets. However, it allows to transfer almost full Ethernet speed on 128 B packets to software. Full line rate transfer can be achieved for 256 B packets using packet trimming or using Unified Headers even for 128 B packets.

The CPU utilization, shown in Figure 6 should be the same for full and trimmed packets for same packet rate. The difference is caused by the fact that higher packet rate can be achieved using trimming. Utilizing the Unified Headers brings the advantage of easier data processing. Therefore, the input thread is always less utilized for the Unified Headers than for trimmed packets. Using the Unified Headers also brings higher performance. The parsing of more complex L3 and L4 headers also causes memory accesses. When it is reduced, more bandwidth is left for the flow cache, which increases the overall performance.

C. Impact of CPU Choice

We have also investigated the impact of the choice of CPU on the packet processing performance. We performed the same tests in the same server with different CPUs. We chose E5-2620 v1 CPUs, which represents a cheaper and therefore slower alternative. Each CPU features six physical cores (12 with hyperthreading), 2.0 GHz operating frequency (2.5 GHz in Turbo mode) and 15 MB of cache. Maximum TDP of each unit is 95 W. The memory controller has throughput of 42.6 GB/s.

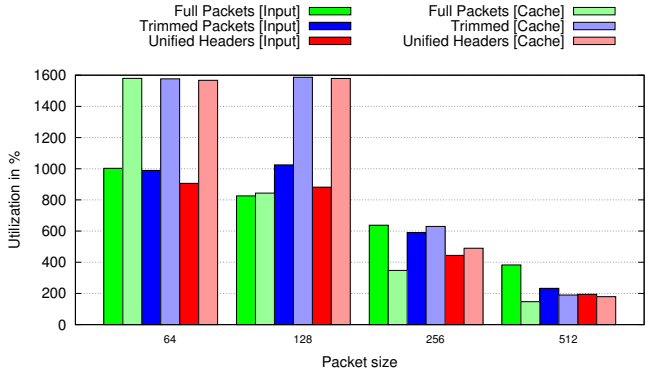

Figure 6. Packet processing comparison using CPU utilization for 16 384 flows per interface.

Figure 7 shows a comparison of the performance for the trimmed packets method. The darker colors are used for the faster CPU. The difference is bigger for smaller number of flows and is reduced with growing utilization of the flow cache. On 64 B packets the performance drop using the slower CPU is 29 % for 128 flows, 23 % for 16 384 flows and 21 % for 131 072 flows.

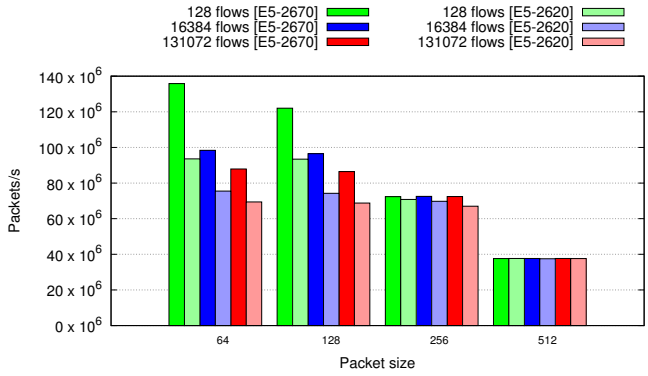

Figure 7. Trimmed packets processing performance comparison for two different CPUs.

We assume that the main difference in the performance is caused by the number of CPU cores. Since there are eight independent DMA channels for the data transfer, the best performance is achieved by eight threads processing the data. However, when only six cores are available, the threads must share the cores and the overhead of switching the threads increases. The difference in the frequency helps mostly for parsing the data, therefore it is useful mainly on full packets. Another bottleneck is caused by the memory controller, since the flow cache updates generate lots of random memory accesses. Using a CPU with wider memory bandwidth helps to alleviate this issue.

VI. Conclusions

The demand for high-density network flow monitoring solutions is increasing. We have built a 2U standard rack server with two COMBO-80G cards. Each card has two 40 G interfaces and can be used in 8×10 G mode. Therefore, the server is theoretically capable of monitoring sixteen 10 G lines, 160 Gbps in total.

We have used a packet generator to fabricate a traffic of different properties and used this traffic to test the capabilities of the monitoring solution. We have worked with different packet lengths and different numbers of active network flows.

Firstly, we have measured the monitoring throughput on full packets, which is a setup that can be achieved using any commodity card with a timestamping unit and an effective distribution among CPU cores. We have shown that the monitoring performance does not achieve the highest possible rate for short packets. However, the maximum rate allowed by the PCI Express bus is achievable for longer packets.

There are several caveats that need to be kept in mind while working with NUMA architecture. Each NIC is connected to one CPU and should store packets in the memory of that CPU. This needs to be enforced by the driver of the NIC. Consequently, the flow exporter for the NIC should also run on the same CPU and work with the corresponding memory. The drivers that we use are not NUMA-aware and the deficiency is clear from the results.

Secondly, we have shown the impact of hardware acceleration on the the flow measurement. Using a packet trimming ensures that the packet rate that can be achieved for short packets applies for longer packets. This technique allows us to monitor full speed 16×10 G Ethernet for 256 B packets. For comparison, the average packet length of our organization's border lines is over 800 bytes. When the packet parsing is performed by the NIC and the information from packet headers are passed using the Unified Headers, the performance is increased even more. The CPU utilization is also lower, since the packet header parsing is a CPU intensive task. However, this approach trades high performance for monitoring flexibility.

The purpose of the last test was to show the difference that can be achieved by using faster CPUs. We have shown that the performance on short packets can rise by more than 30 % when using faster CPUs with higher memory bandwidth. We conclude that the monitoring system can achieve even higher performance by utilizing better CPUs. However, a cost to performance ratio is also important for a production use.

There are several improvements that can be made to achieve higher performance. We can buy better CPUs, but the budget for monitoring systems is often limited. The driver for the NICs should be made NUMA-aware to avoid costly memory accesses using QPI bus. We have also shown that features like packet trimming can significantly improve the performance of flow monitoring systems. If the next generation commodity cards should be used for the network flow monitoring, such features can turn out to be crucial for handling large volumes of data. The performance of the PCI Express bus can be doubled by using 16 lanes, which is an approach that can be expected to be used in the future.

Acknowledgement

This material is based upon work supported by the project TA03010561 funded by the Technology Agency of the Czech Republic.

References

[1] B. Claise, "Cisco Systems NetFlow Services Export Version 9," RFC 3954, Internet Engineering Task Force, October 2004.

[2] B. Claise, B. Trammell, and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information," RFC 7011, Internet Engineering Task Force, Sep. 2013.

[3] L. Deri, "Passively Monitoring Networks at Gigabit Speeds Using Commodity Hardware and Open Source Software," in *In Passive and Active Measurement Workshop 2003. NLANR/MNA*, 2003.

[4] L. Degioanni and G. Varenni, "Introducing Scalability in Network Measurement: Toward 10 Gbps with Commodity Hardware," in *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '04. New York, NY, USA: ACM, 2004, pp. 233–238. [Online]. Available: http://doi.acm.org/10.1145/1028788.1028818

[5] R. Clegg, M. Withall, A. Moore, I. Phillips, D. Parish, M. Rio, R. Landa, H. Haddadi, K. Kyriakopoulos, J. Auge, R. Clayton, and D. Salmon, "Challenges in the capture and dissemination of measurements from high-speed networks," *Communications, IET*, vol. 3, no. 6, pp. 957–966, June 2009.

[6] L. Braun, A. Didebulidze, N. Kammenhuber, and G. Carle, "Comparing and Improving Current Packet Capturing Solutions Based on Commodity Hardware," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '10. New York, NY, USA: ACM, 2010, pp. 206–217. [Online]. Available: http://doi.acm.org/10.1145/1879141.1879168

[7] F. Fusco and L. Deri, "High Speed Network Traffic Analysis with Commodity Multi-core Systems," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '10. New York, NY, USA: ACM, 2010, pp. 218–224. [Online]. Available: http://doi.acm.org/10.1145/1879141.1879169

[8] L. Deri *et al.*, "Improving passive packet capture: Beyond device polling," 2004. [Online]. Available: http://luca.ntop.org/Ring.pdf

[9] G. Antichi, S. Giordano, D. Miller, and A. Moore, "Enabling open-source high speed network monitoring on NetFPGA," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*, April 2012, pp. 1029–1035.

[10] L. Deri, A. Cardigliano, and F. Fusco, "10 Gbit Line Rate Packet-to-disk Using n2disk," in *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*, April 2013, pp. 441–446.

[11] J. García-Dorado, F. Mata, J. Ramos, P. Santiago del Río, V. Moreno, and J. Aracil, "High-Performance Network Traffic Processing Systems Using Commodity Hardware," in *Data Traffic Monitoring and Analysis*, ser. Lecture Notes in Computer Science, E. Biersack, C. Callegari, and M. Matijasevic, Eds. Springer Berlin Heidelberg, 2013, vol. 7754, pp. 3–27. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-36784-7_1

[12] R. Hofstede, P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow Monitoring Explained: From Packet Capture to Data Analysis with NetFlow and IPFIX," *Communications Surveys Tutorials, IEEE*, 2014.

[13] Intel Corporation, "Intel Ethernet Converged Network Adapters XL710 10/40 GbE," online, 2014. [Online]. Available: http://www.intel.com/content/www/us/en/network-adapters/converged-network-adapters/ethernet-xl710.html

[14] INVEA-TECH a.s., "COMBO-80G FPGA Card," online, 2014. [Online]. Available: https://www.invea.com/en/products-and-services/fpga-cards/combo-80g

[15] INVEA-TECH a.s., "FlowMon Probe," online, 2014. [Online]. Available: https://www.invea.com/en/products-and-services/flowmon/flowmon-probes

[16] Intel Corporation, "An Introduction to the Intel QuickPath Interconnect," online, 2009. [Online]. Available: http://www.intel.com/content/dam/doc/white-paper/quick-path-interconnect-introduction-paper.pdf